Validating Polynomial Numerical Computations with Complementary Automatic Methods

Philippe Langlois, Nathalie Revol

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Validating polynomial numerical computations with complementary automatic methods

Philippe Langlois, INRIA

Nathalie Revol , INRIA

No 4205

Juin 2001

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4205--FR+ENG

RHÔNE-ALPES

Validating polynomial numerical computations with complementary automatic methods

Philippe Langlois, INRIA
Nathalie Revol , INRIA

Thème 2 — Génie logiciel
et calcul symbolique
Projet Arénaire

Rapport de recherche n ° 4205 — Juin 2001 — 14 pages

Abstract: Finite precision computations affect the accuracy of computed solutions and sometimes the stability of iterative algorithms. Automatic approaches exist to control and to reduce these effects. Examples are the CESTAC and the CENA methods and the more general interval approaches. We focus here on a complementary use of these methods to localize unstable behavior of the algorithm, to improve the accuracy of the solutions, to identify and explain finite precision effects. We present computational experiments on ill-conditioned polynomial roots approximated with Newton's iteration that illustrate the well-known influence of coefficient perturbations.

Key-words: Finite precision, automatic rounding error analysis, Newton's iteration, polynomial multiple roots.

(Résumé : tsvp)

This text is also available as a research report of the Laboratoire de l'Informatique du Parallélisme `http://www.ens-lyon.fr/LIP` and extends previous results presented by one of the authors at the 16th IMACS World Congress (Lausanne, August 2000).

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
Téléphone : 04 76 61 52 00 - International: +33 4 76 61 52 00
Télécopie : 04 76 61 52 52 - International: +33 4 76 61 52 52

Validation de calculs numériques polynomiaux par des méthodes automatiques complémentaires

Résumé : Les calculs en précision finie peuvent affecter la précision des solutions calculées et parfois même la stabilité des algorithmes itératifs. Des approches automatiques existent pour contrôler et réduire ces effets. Des exemples sont la méthode CESTAC, la méthode CENA et les approches plus générales avec des intervalles. Nous nous intéressons ici à une application complémentaire de ces méthodes pour localiser le comportement instable de l'algorithme, améliorer la précision des solutions, identifier et expliquer les effets de la précision finie. Nous présentons des expérimentations numériques sur le calcul par la méthode de Newton de racines polynomiales mal conditionnées qui illustrent l'influence bien connue des perturbations des coefficients.

Mots-clé : Précision finie, analyse d'erreur automatique, itération de Newton, racine multiples de polynômes.

1 Introduction

Finite precision corrupts fundamental properties of numerical algorithms that hold in exact arithmetic. The stability of the algorithm or the accuracy of the computed solution may be affected by floating point computation. A well-known example is the computation with Newton's iteration of ill-conditioned roots of polynomial, *i.e.*, multiple or cluster of roots [1]. Rounding errors analysis is principally performed "by hand" [2] but automatic approaches use the computer to provide more general and easy-to-use applications.

In this paper, we focus on three automatic approaches : the CESTAC, the CENA and an interval methods. The CESTAC method, developed by J. Vignes and his colleagues since 1974, is a forward stochastic method [3]. It provides a dynamic estimation of the number of significant digits of computed results that allows instability localization and accuracy control. The CENA method aims to automatically improve the accuracy of the computed result. This method computes both a linear correcting term and a confidence bound for the corrected value of a specific class of algorithms [4]. Improving the accuracy of sensitive intermediate variables may also stabilize algorithms suffering from rounding errors. Interval methods provide guaranteed enclosures for the results; yet they do not sharply estimate the loss of accuracy since pessimistic overestimations are usually returned. The use of a multiple precision floating point arithmetic, combined with interval arithmetic, allows to reduce this overestimation and thus aim to provide an arbitrary accuracy for the guaranteed results.

We experiment these methods considering an ill-conditioned root of a polynomial computed with Newton's iteration. As our computations are restricted to a unique polynomial, the results we present here should not be interpreted as an assessment of the respective quality of the different methods. This paper aims to illustrate how these automatic methods help to identify and to explain finite precision effects.

We briefly review the different methods in Section 2 and the characteristics of floating point computation of polynomial ill-conditioned roots in Section 3. In Section 4, we describe our experiments and analyze the results to highlight finite precision effects.

2 Stochastic and Deterministic (Point and Interval) Methods

We limit the following presentation to the main and hereafter useful aspects of the CESTAC and CENA methods and of interval arithmetic. The description of the main characteristics of the CESTAC method is faithful to the presentation by its authors and colleagues, as for example J. Vignes and J.-M. Chesneaux in [3, 5]. The version of the CADNA library we use hereafter is described in [6]. The main reference for the CENA method is [4]. Introductory references to interval arithmetic and methods are [7, 8, 9].

2.1 The CESTAC Method

The CESTAC method simulates discrete stochastic arithmetic (DSA) to provide an estimation of the number of significant digits in a computed result. The CESTAC method applies the following scenario to the floating point operations of any numerical algorithm.

1. Two or three synchronous runs of the operation are performed with a specific random arithmetic to provide a sample of computed results R_i, $i = 1, \ldots, N$ with $N = 2$ or 3.
2. The mean value of the sample (R_i), $R = (1/N) \sum_{i=1}^{N} R_i$, is the computed result.
3. The number C_R of significant digits in R is estimated with the Student distribution associated to R, computing

$$C_R = \log_{10} \left(\frac{\sqrt{N}|R|}{s\tau_\beta} \right), \text{ with } s^2 = \frac{1}{N-1} \sum_{i=1}^{N} (R_i - R)^2, \tag{1}$$

and τ_β is the value of the Student distribution of $(N-1)$ degrees of freedom and a probability level $1 - \beta$.

This statistical approach assumes that the elementary rounding errors, *i.e.*, the rounding errors each floating point operation generates, are independent and identically distributed random variables, and that their first order effect predominates in the global error, *i.e.*, in the accuracy of the computed result. With these hypotheses, the authors prove that the sample (R_i) provides a statistical modeling (as a centered Gaussian

random variable) of the exact result r. They derive relation (1) that yields the estimated number C_R of significant decimal digits shared by r and R.

This estimation allows the definition of the value $\underline{0}$, named computational zero in CESTAC and stochastic zero in DSA, that verifies $R_i = 0$ $(i = 1, \ldots, N)$ or $C_R \leq 0$. This specific value $\underline{0}$ describes a zero or a non-significant computed value, *i.e.*, a value with no significant digit guaranteed by the method. With this zero, the authors define stochastic extensions of the equality and order relations. This discrete stochastic arithmetic formalizes these relations when only the significant guaranteed digits of the operands are considered. We will consider another practical use of this zero that defines corresponding stopping criteria for iterative processes in Section 3.2.

The CADNA library implements the CESTAC method and consists in computing the sample R_i using a random distribution of the different rounding modes of the floating point arithmetic. The use of the Fortran 90 and Ada implementations of the CADNA library is straightforward since overloaded operators implement the stochastic types and DSA. The CADNA library also dynamically controls the validity of the first order predominance hypothesis testing the guaranteed accuracy of the multiplication operands and the denominators ; this control provides a self-validation of the CESTAC method. Other dynamic controls of the same kind localize the operations or the tests that suffer from a significant loss of accuracy. These facilities allow the user to perform a numerical debugging of the computation.

2.2 The CENA Method

The CENA method computes a correcting factor that generally improves the accuracy of a computed result. This correction relies on the first order approximation of the global error, algorithmic differentiation and the computation of elementary absolute rounding errors. Let $\widehat{x}$ be a scalar result computed with floating point arithmetic and x_i, $i = n+1, \ldots, N$, be the intermediate variables introduced to evaluate $\widehat{x} = x_N$. The method yields the corrected result $\overline{x}$ defined as

$$\overline{x} = \widehat{x} + \widehat{\Delta}_L(x), \tag{2}$$

with

$$\widehat{\Delta}_L(x) = \sum_{i=n+1}^{N} \frac{\partial \widehat{x}}{\partial \delta_i} \cdot \delta_i.$$

This correcting factor $\widehat{\Delta}_L$ is the computed linearization of the global error in $\widehat{x}$ with respect to the elementary errors δ_i, $i = n+1, \ldots, N$, that are the rounding errors generated by the intermediate computations of $\widehat{x}$. Algorithmic differentiation yields the partial derivatives in $\widehat{\Delta}_L$ (see [10] for details on algorithmic differentiation). Elementary errors are estimated with specific floating point algorithms for operators $+, -, \times, /, \sqrt{}$ (see [11, 12, 4] for explicit relations).

As the correcting factor $\widehat{\Delta}_L$ and the correction (2) also suffer from rounding errors, a validity bound B_{E_C} of the corrected result is computed. This dynamic bound derives from a running error analysis of the algorithmic differentiation and the correction process (see [2] for running error analysis).

Therefore, the scalar error-free result x satisfies $x \in [\overline{x} - B_{E_C}, \overline{x} + B_{E_C}]$ assuming that the first order approximation of the global error is valid. This hypothesis is satisfied for linear algorithms as defined in [13, 4].

Evaluating polynomial by Horner's scheme is a linear algorithm and so motivates the use of the CENA method in our experiments. In Section 4.4, we describe a corrected version of Newton's iteration that relies on intermediate corrections. The direct application of the CENA method to the final result of a computation improves its accuracy. Such a final correction is not interesting here as the considered algorithms are iterative. Other previous results have illustrated that the more general correction of sensitive intermediate variables also improves the stability of this iteration when it is applied to compute ill-conditioned roots [14].

Current implementations of the CENA method use overloading facilities of Fortran 90 and Ada to facilitate its application when IEEE-754 arithmetic is available.

2.3 Interval arithmetic

The fundamental principle of interval arithmetic consists in enclosing every number by an interval containing it and being representable by machine numbers : for instance it can be stored as its lower and upper extremities and these bounds are machine numbers. For example, on a radix-10 machine with 3 digits of mantissa, π would be represented by the interval $[3.14, 3.15]$. The middle of the interval $X = [\underline{x}, \overline{x}]$ will be denoted by $\mathrm{mid}(X) = (\underline{x} + \overline{x})/2$ and the width of X by $w(X) = \overline{x} - \underline{x}$. Capital letters will stand for intervals and small letters for scalars. The arithmetic operations are extended for interval operands to guarantee the error-free results belong to the computed interval

$$X \diamond Y = \{x \diamond y \ : \ x \in X, \ y \in Y\},$$

where X and Y denote interval operands and $\diamond = +, -, \times$ or $/$. The definitions of the usual functions can also be extended to include interval arguments : for instance $\sqrt{[5, 8]} = [\sqrt{5}, \sqrt{8}]$ since $\sqrt{\ }$ is an increasing function, and $\sin[\pi/3, \pi] = [0, 1]$, since care must be taken for sin which is not monotonous.

As floating point arithmetic, interval arithmetic loses some algebraic properties, in particular the $-$ operator is no more the reciprocal of $+$ and the multiplication is only subdistributive over the addition.

When operands are replaced by intervals in an expression, such as a polynomial one, the computed result is an interval enclosing every possible value of this expression for any scalar value in the input intervals. This *guaranteed result* is the main advantage of interval arithmetic. However, this guaranteed interval can be too large and depends on the chosen form for the evaluated expression.

The two main origins of this overestimation of the results are the well-known decorrelation of the variables and the wrapping effect. The former, also known as dependency problem, appears when computing $X \times X$ as $\{x \times y : x \in X, y \in X\}$ for which the (identity) correlation between x and y is lost. The wrapping effect turns out in $\mathbb{R}^n$ where the result must be a rectangular box with sides parallel to the axes enclosing an oblique and possibly very thin set. In order to limit this swell, Horner's scheme is preferred among other expressions to evaluate a polynomial. This scheme provides tighter results than other evaluations based on an expanded expression – the proof relies on the subdistributivity of $\times$ over $+$ [9, 15]. If $p(x) = a_0 + a_1 x + \ldots a_n x^n$, the interval Horner scheme can be written as

$$P_H(X) = a_0 + X \times (a_1 + X \times (\ldots + a_n \times X) \ldots) \supset p(X).$$

The use of Taylor expansions of order 1 or 2 is also classical to limit the overestimation of evaluation, providing tighter bounds when the input interval is small. The mean-value theorem, based on Taylor expansion of order 1, is interpreted in interval arithmetic as

$$P_{T1}(X) = P_H(\{\mathrm{mid}(X)\}) + P'_H(X) \times (X - \mathrm{mid}(X)) \supset p(X),$$

where P'_H is the interval Horner evaluation of p'. Similarly the second order Taylor expansion leads to

$$\begin{aligned} P_{T2}(X) = P_H(\{\mathrm{mid}(X)\}) + P'_H(\{\mathrm{mid}(X)\}) \times (X - \mathrm{mid}(X)) \\ +1/2 P''_H(X) \times (X - \mathrm{mid}(X))^2 \supset p(X). \end{aligned}$$

Accurate results are computed evaluating the polynomials both with Horner scheme and Taylor expansions : in the experiments presented below we calculate the interval

$$P(X) = P_H(X) \cap P_{T1}(X) \cap P_{T2}(X) \supset p(X).$$

Let us precise that the polynomials are symbolically differentiated.

The interval arithmetic library used in our experiments is the MPFI library *Multiple Precision Floating-point Interval library*. It is based on MPFR (*Multiple Precision Floating-point Reliable library*), a library for multiple precision arithmetic providing correct roundings (downward, upward, to nearest and toward zero) designed on top of GMP and intending at replacing the semantically not so well founded MPF in GMP ; MPFR and MPFI are developed at INRIA [16, 17].

3 Computing Ill-Conditioned Roots of Polynomial

Finite precision effects on the computation of ill-conditioned roots of polynomial have been numerously analyzed in the past, *e.g.*, see [18] for entries. We present here the main useful properties for our purpose.

3.1 Ill-Conditioning and Attainable Accuracy

Ill-conditioned roots are sensitive to small perturbations of the polynomial coefficients. Such coefficient perturbations may be data errors, representation errors, *e.g.*, decimal to binary conversion, and/or may come from the backward effect of rounding errors in the algorithm provided by the backward analysis. In both cases, the computation may yield inaccurate roots and/or a wrong number of computed roots and hence with arbitrary order.

The following property highlights the sensitivity of polynomial roots to perturbations [19]. Let x^* be a root of multiplicity m of polynomial $p(x) = a_n x^n + \cdots + a_0$. A relative error $\mathbf{u}$ of a_i perturbs the root x^* to $x^*(\mathbf{u})$ such that the forward error satisfies a first order approximation

$$x^*(\mathbf{u}) - x^* = \mathbf{u}^{1/m} \left[-\frac{m!\, a_i x^{*i}}{p^{(m)}(x^*)} \right]^{1/m}. \tag{3}$$

Hence, multiple roots are always ill-conditioned and the forward error is of the order of $\mathbf{u}^{1/m}$. Single roots are also ill-conditioned when $\left| a_i x^{*i} / p'(x^*) \right| \gg \mathbf{u}$.

Let us remark that relation (3) yields the attainable accuracy, *i.e.*, the best forward error bound we can expect when the root finding method evaluates $p(x)$, *e.g.*, Newton's method. Let $\delta(p,x)$ be the absolute rounding error in $p(x)$ evaluation, $\widehat{p}(x) = p(x) + \delta(p,x)$, and δ such that $|\delta(p,x)| \leq \left| a_{i_0} x^{*i_0} \right| \leq \delta$ $(0 \leq i_0 \leq n)$. The best computed approximation $\widehat{x}^*$ of x^* satisfies $\widehat{p}(\widehat{x}^*) = 0$. From relation (3), the attainable accuracy is

$$|\widehat{x}^* - x^*| \leq \left(\frac{m!\,\delta}{|p^{(m)}(x^*)|} \right)^{1/m}. \tag{4}$$

As in [20], we remark that relation (4) is a method-independent error estimate. The accuracy of the polynomial evaluation controls the accuracy of the computed root. We discuss the practical use of this attainable accuracy in the next paragraph.

3.2 Newton's Iteration, Finite Precision and Stopping Criteria

Assuming that the first iterate $x(0)$ is close enough to the root x^*, we compute the Newton iteration

$$x(k+1) = x(k) - \frac{p(x(k))}{p'(x(k))}. \tag{5}$$

In exact arithmetic, iteration (5) converges quadratically to a single root and only geometrically with ratio $m/(m-1)$ to a root of multiplicity m [21]. In finite precision, the convergence of iteration (5) may suffer from the following limitations (computed quantities have lost the hat hereafter).

1. $p'(x(k)) = 0$ but $p(x(k)) \neq 0$,
2. $x(k+1) = x(k)$ but $p(x(k)) \neq 0$,
3. $p(x(k)) = p'(x(k)) = 0$ but $x(k)$ is not an accurate approximation of x^*.

Hence, a reliable stopping criterion of iteration (5) has to avoid such cases. The previously discussed attainable accuracy should be useful to define such a criterion. Alas, relation (4) is not computable as long as the multiplicity of the root remains unknown, and that is often the case.

In practice, the following criteria are well-known for terminating iteration (5).

RE (Relative Evolution) : $|x(k+1) - x(k)| \leq \mathbf{u}' |x(k)|$;

AR (Absolute Residual) : $|p(x(k+1))| \leq \sigma'$.

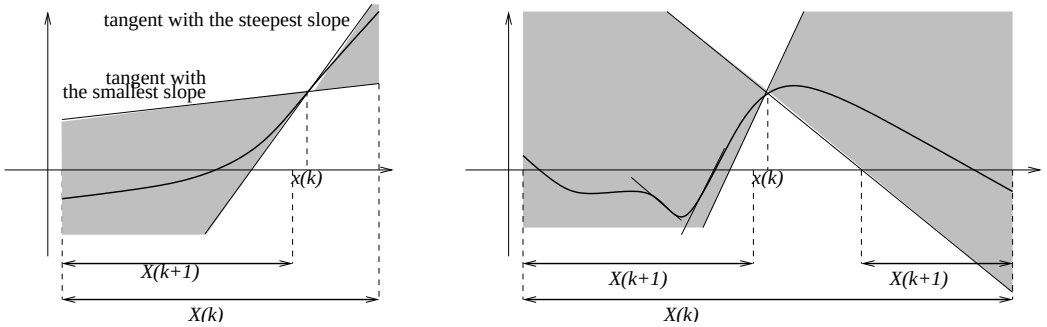

FIG. 1: Interval Newton iteration.

The *a priori* relative bound $\mathbf{u}' = O(\mathbf{u})$ limits the stagnation of the iterate but is generally inefficient for ill-conditioned roots. Choosing $\sigma' = O(\sigma)$ where σ is the smallest non-zero positive floating point number ($\sigma \ll \mathbf{u}$) would be possible if the computation of the residual $p(x(k+1))$ were error free. Considering relation (4), σ' should dynamically control the inaccuracy of the polynomial evaluation, choosing for example $\sigma' = \mathbf{u}\left(\sum_{i=0}^{n} |a_i\, x(k+1)^i|\right)$ [22].

With stochastic arithmetic, it is well-known that the following alternative criteria are available [3].

SAE (Stochastic Absolute Evolution) : $|x(k+1) - x(k)| = \underline{0}$;

SAR (Stochastic Absolute Residual) : $p(x(k+1)) = \underline{0}$.

These criteria use the specific value $\underline{0}$. We recall it describes a zero value or a non-significant computed value [23, 24]. When $x(k+1)$ satisfies the SAE test, computing $p(x(k+1))$ is necessary to decide whether $x(k+1)$ is an acceptable approximation of the root.

3.3 Interval Newton Iteration

The naive interval iteration $X(k+1) = X(k) - P(X(k))/P'(X(k))$ derived from (5) by replacing scalar iterates and polynomial evaluations by their interval counterparts diverges since the width of iterated interval increases as $w(X(k+1)) = w(X(k)) + w(P(X(k))/P'(X(k)))$.

The principle of the interval Newton iteration consists in enclosing the graph of the function p on $X(k)$ in a cone $C(X(k), x(k))$ [8]. This cone $C(X(k), x(k))$ is defined by the point $M(k) = (x(k), p(x(k)))$ where $x(k)$ belongs to $X(k)$ and the two straight lines through $M(k)$ whose slopes are the steepest and the less steep tangents of p over $X(k)$. This region is the gray zone on Figure 1 and the extremal tangents are represented on the right part. In practice, $x(k) = \mathrm{mid}(X(k))$ is a classical choice and the extremal values of p' over $X(k)$ are replaced by the extremities of $P'(X(k))$, the interval evaluation of p' over $X(k)$. The roots of p belonging to $X(k)$ are enclosed in the intersection of this cone with the x-axis. We take the intersection of this set with $X(k)$ in order to limit the search to the interval $X(k)$. The corresponding next iterate $X(k+1)$ satisfies

$$X(k+1) = \left(x(k) - \frac{P(\{x(k)\})}{P'(X(k))}\right) \cap X(k), \tag{6}$$

where P (resp. P') is an interval function corresponding to p (resp. p').

The scheme of this iteration is illustrated on Figure 1 : on the left $X(k+1)$ consists in one interval, on the right two sub-intervals of $X(k)$ are created, each one containing at least one root in this example. More generally, an interval algorithm outputs a list of intervals that may contain a solution, denoted by Res in algorithm 1 below. Consequently, an interval algorithm handles a list, denoted by $\mathcal{L}$, of intervals waiting to be processed.

The following stopping criteria are used for interval Newton iteration :

RRA (Relative Root Accuracy) : $w(X(k+1))/|\mathrm{mid}(X(k+1))| \leq \mathbf{u}_X$;

ARA (Absolute Residual Accuracy) : $w(P(X(k+1))) \leq \mathbf{u}_Y$.

These stopping criteria are adapted from RE and AR. The absolute residual accuracy is straightforwardly adapted for interval Newton iteration. The choice of $\mathbf{u}_Y$ is motivated as for the AR criterion. The ideal one would be $\mathbf{u}_Y = O(\sigma)$, *i.e.*, about the smallest positive floating point number, if the computation of $P(X(k+1))$ were error-free and $w(X(k+1))$ were arbitrarily small. Nevertheless rounding errors and interval overestimations enforce to chose a much larger $\mathbf{u}_Y$ in the experiments presented below. The RRA criterion, corresponding to RE, measures the relative accuracy achieved on the result measured by the relative width of $X(k+1)$; it has to be smaller than a prescribed value $\mathbf{u}_X$ which can be as small as $O(\mathbf{u})$.

In order to ensure the termination of the interval Newton method, the iteration has to be modified so that it always reduces the width $w(X(k+1))$ by (at least) a constant factor α compared to $w(X(k))$. This is achieved by replacing the Newton step by a dichotomy step if the Newton step yields a too large $X(k+1)$. This is implemented by algorithm 1 we comment now.

Input : P, P', X_0 { `the initial search interval` }
  $\mathcal{L} = \{X_0\}$, $\alpha = 0.75$ { `any value in` $]0.5, 1[$ `is suitable` }
2: **while** $\mathcal{L} \neq \emptyset$ **do**
  Suppress $(X, \mathcal{L})$
4: $x := \text{mid}(X)$
  $(X_1, X_2) := (x - \frac{P(\{x\})}{P'(X)}) \cap X$ { X_1 `and` X_2 `can be empty` }
6: **if** $w(X_1) > \alpha w(X)$ or $w(X_2) > \alpha w(X)$ **then**
    $(X_1, X_2) := bisect(X)$
8: **end if**
  if $X_1 \neq \emptyset$ and $P(X_1) \ni 0$ **then**
10: **if** RRA or ARA **then**
      Insert X_1 in Res
12: **else**
      Insert X_1 in $\mathcal{L}$
14: **end if**
  end if
16: { `same handling of` X_2 }
 end while
Output : Res { `a list of intervals that may contain the roots` }

Algorithm 1: Interval Newton's iteration

1. The starting interval is given by the user and the algorithm explores every sub-interval X such that the interval evaluation $P(X)$ contains 0 (line 9). The intervals to be examined are stored in a working list $\mathcal{L}$ (line 13).
2. An interval X accepted as a potential result is inserted in the *Res* list (line 11). Two tests (not explained in the algorithm) are performed on X : using the intermediate value theorem, the existence of a root in X can be proven[1] and using a theorem valid for interval Newton iteration only [8], the existence and uniqueness of a root can be proven (uniqueness also means that it is a single root). If none of these tests applies, the interval X is kept as a *putative* result, *i.e.*, it may contain a root.
3. The *bisect* operation splits an interval X into two halves X_1 and X_2 such that $X_1 \cup X_2 = X$ and $w(X_1) = w(X_2) = w(X)/2$ (line 7).

[1] These results are equivalent to the ones annotated as `SAFE` by Numerica [25].

The two previously mentioned tests for existence and existence and uniqueness are more likely to give a positive answer when $w(X(k+1))$ and thus $w(P(X(k+1)))$ are small : indeed Taylor expansions yield tight results on tight inputs, cf. section 2.3. Quite often, the fixed IEEE-754 precisions forbid to satisfy the RRA test for value of $\mathbf{u}_X$ arbitrarily smaller than the fixed precision $\mathbf{u}$. Multiple precision computations enable to implement an arbitrarily accurate stopping criteria as it will be shown in Section 4.7. With an automatic adaptation of the precision (not detailed here), we increase the precision until both criteria RRA and ARA are satisfied [26]. In that case, $\mathbf{u}_X$ and $\mathbf{u}_Y$ are *a priori* chosen by the user.

4 Experimental Results

Let us consider the computation of the double root $x^* = 3/7$ of polynomial

$$p(x) = 1.47x^3 + 1.19x^2 - 1.83x + 0.45;$$

the other root is $y^* = -5/3$ [5]. Numerical experiments are performed using IEEE-754 binary floating point arithmetic on a Sun Ultra5 workstation (SunOS 5.7), FORTRAN 90 (WorkShop Compilers 5.0, FORTRAN 90 2.0), GNU C compiler (version 2.95.2, option -O2), CADNA library v2.2, MPFR 2001 and MPFI (April 2001).

4.1 Scenario of experiments

We limit the discussed experiments to IEEE-754 single precision results, *i.e.*, $\mathbf{u_s} \approx 5.96 \times 10^{-8}$, except in the next paragraph where are also proposed double precision results, *i.e.*, $\mathbf{u_d} \approx 1.11 \times 10^{-16}$.

We analyze the following numerical experiments.

1. The classic computation with IEEE-754 arithmetic of Newton's iteration yields reference results in single (and double) precision.
2. The CADNA library and its associated stopping criteria are applied to the iteration and provide a result with guaranteed significant digits.
3. The corrected iteration with the CENA method is initialized with the previous result to compute more accurate approximate roots.
4. Fixed precision interval computations are performed considering the polynomial coefficients as
   (a) scalars being the single precision rounded (to the even nearest) floating point numbers,
   (b) intervals whose endpoints are the downward and upward rounded floating point numbers.
5. Exact computations to validate the previously computed results are performed with Maple.
6. The interval experiments of (4) are performed again using adaptative multiple precision.

4.2 The IEEE-754 Computation

We compute Newton iteration (5) in IEEE-754 single and double precisions using Horner's scheme for the polynomial evaluations $\widehat{p}$ and $\widehat{p}'$. The two stopping criteria RE and AR are implemented respectively with $\mathbf{u}' = \mathbf{u_s}$ or $\mathbf{u_d}$ and $\sigma' = 1.0 \times 10^{-38}$. Starting with $x(0) = 0.5$, both computations yield $x(k)$ satisfying the stopping criterion AR (see next table). Other acceptable choices of $x(0)$ give similar results.

Precision	$x(0)$	k	$x(k)$	$p(x(k))$	$p'[x(k)]$	$\frac{\lvert x^* - x(k)\rvert}{\lvert x^*\rvert}$
$\mathbf{u_s}$	0.5	16	0.4285 6118	0.0	-6.31E-5	2.39E-5
$\mathbf{u_d}$	0.5	25	0.4285 7143 3545 3048	0.0	3.06E-8	1.67E-9

The absolute error bound for polynomial p evaluated by Horner's scheme in a root neighbourhood satisfies $\delta \approx \mathbf{u}'$. Hence comparing with $x^* = 0.4285\,7142\,86$, the computed root $x(k)$ agrees with the predicted accuracy given by relation (4) for a root of multiplicity 2.

4.3 Localizing the instability with the CESTAC Method

We use the CADNA library to compute iteration (5) with criteria SAE and SAR. Starting again with $x(0) = 0.5$, both SAE and SAR criteria are satisfied for the following value.

Precision	$x(0)$	k	$x(k)$	$p(x(k))$	$p'[x(k)]$
$\mathbf{u_s}$	0.5	10	**0.4287**	0	0.96E-3

The CESTAC method reduces the number of iterations and provides 3 guaranteed decimal digits (CADNA displays only significant digits and does not guarantee the last displayed decimal digit [6]). To analyze the forward error, we remark that the guaranteed digits of the CADNA result are exact comparing to x^*. The number of significant digits agrees with the predicted accuracy.

4.4 Improving the Accuracy with the CENA Method

The corrected iteration (5) with the CENA method is

$$x(k+1) = x(k) - \frac{\overline{p}[x(k)]}{\overline{p'}[x(k)]} \tag{7}$$

where $\overline{p} = \widehat{p} + \widehat{\Delta}_L(p)$ and $\overline{p'} = \widehat{p'} + \widehat{\Delta}_L(p')$ are the corrected evaluations of polynomials p and p'. Iteration (7) is controlled with the stopping criteria RE and AR. The B_{E_C} denotes the dynamically computed bound for $\overline{p} - p$.

Using CADNA result, an optimal initial value $x(0)$ satisfies $x(0)_- \leq x(0) \leq x(0)_+$ with the IEEE-754 single precision values $x(0)_- = 0.4280\,0000$ and $x(0)_+ = 0.4289\,9999$. We compute iteration (7) using the two initial values $x(0)_-$ and $x(0)_+$; the results are the following.

Precision	$x(0)$	k	$x(k)$	$p(x(k))$	$\frac{\lvert x(k)-x(k-1)\rvert}{\lvert x(k)\rvert}$	B_{E_C}
$\mathbf{u_s}$	$x(0)_-$	7	0.4284 9594	3.70E-12	0.0	2.60E-14
$\mathbf{u_s}$	$x(0)_+$	6	0.4286 4692	4.83E-12	0.0	9.19E-15

Both iterations terminate satisfying the RE criterion. So no more accurate approximation of the root could be computed in IEEE-single precision starting with chosen $x(0)$. The two computed approximations $\overline{x}^*_- = 0.4284\,9594$ and $\overline{x}^*_+ = 0.4286\,4692$ are different after the third digit. The B_{E_C} bound and the corrected residual $\overline{p}(x(k))$ values prove that none of the two computed approximates is a root of the corrected polynomial $\overline{p}$. This polynomial $\overline{p}$ has two separate roots in the neighbourhood of x^* approximated by $\overline{x}^*_-$ and $\overline{x}^*_+$. Other choices of $x(0)$, $x(0)_- \leq x(0) \leq x(0)_+$, yield similar convergence to the same two finite precision limits $\overline{x}^*_-$ and $\overline{x}^*_+$.

4.5 Fixed precision for interval Newton iteration

4.5.1 Polynomial coefficients are scalars

The interval Newton iteration has been experimented using a fixed precision corresponding to the IEEE single precision. In this series of experiments the polynomial coefficients are handled as scalars and not intervals : they are the rounded to nearest floating-point numbers corresponding to the given coefficients. The starting search interval is $[-2, 2]$: the existence and uniqueness of the root corresponding to $-5/3$ is proven and will not be mentioned any more. The required (relative) accuracy on the root is 10^{-6}. Different values for the (absolute) accuracy on the residual have been tested, we present two typical results below.

Case 1 : when the absolute residual $\mathbf{u}_Y$ is quite large, then the ARA test constitutes the effective stopping criterion, since it is easily fulfilled.

Case 2 : when the absolute residual $\mathbf{u}_Y$ is smaller, then the effective stopping criterion is given by the relative accuracy of the roots., *i.e.*, the RRA test.

	$\mathbf{u}_X$	$\mathbf{u}_Y$	putative roots	roots with proven existence
Case 1	10^{-6}	10^{-6}	[0.4283 6588, 0.4289 9442]	none
Case 2	10^{-6}	10^{-10}	[0.4284 2531, 0.4287 0445]	[0.4284 7672, 0.4285 2819]
				[0.4286 1747, 0.4268 7646]

In both cases, the results are accurate up to the third decimal digit and comply with the result given by CADNA.

In case 1, the results obtained with the CENA method both belong to this interval, as well as the result obtained with the IEEE-754 computations.

In case 2, the two intervals are guaranteed to contain (at least) one root. Each of these two intervals contains a result provided with CENA, however none contains the result obtained by the IEEE computations, *i.e.*, the IEEE result cannot be a root of the considered polynomial. Since the input polynomial is of degree 3, then we have obtained enclosures for every root, which are thus single roots (we recall that the other root is out of the scope of the present discussion). This transformation of a double root into two single roots is explained in Section 4.6.

4.5.2 Polynomial coefficients are intervals

Now each coefficient of the polynomial is stored as an interval whose endpoints are the original coefficients, rounded downward and upward. The meaning of a discarded interval is that it cannot contain a root of a polynomial whose coefficients belong to the interval coefficients.

When $\mathbf{u}_X = 10^{-6}$ and $\mathbf{u}_Y = 10^{-6}$, the result is the interval $[0.4283\,7473, 0.4289\,963]$: it contains all the results previously obtained in this section. When $\mathbf{u}_X = 10^{-6}$ and $\mathbf{u}_Y = 10^{-10}$, a list of 19 putative root(s) enclosures is returned, covered by $[0.4283\,8624, 0.4287\,4685]$.

The results obtained by direct IEEE computations, by CADNA and by the CENA method are again enclosed in these intervals.

With these results, we can conclude nothing about the existence and the uniqueness of a root. Possible explanations are either that the overestimations are too severe or that really nothing can be concluded. The use of a multiple precision arithmetic in Section 4.7 will remove the overestimation problem.

4.6 The Exact Computed Solution

These experiments actually illustrate a well-known effect of finite precision when computing multiple polynomial roots. The double root x^*, *i.e.*, the exact arithmetic solution, is transformed into two separate single roots $\widehat{x}^*_-$ and $\widehat{x}^*_+$ in finite precision. Nevertheless, the convergence rate and the attainable accuracy are controlled by the order 2 of the exact problem root.

IEEE-754 computation suffers from the inaccurate polynomial evaluation $\widehat{p}(x)$ and no guarantee could be given for digits after about half the precision (as relation (4) indicates). The CESTAC method localizes this instability and returns these significant digits. In both cases, the attainable accuracy bound prevents to distinguish these separate roots. Computing corrected $\overline{p}$, the CENA method improves the accuracy of the polynomial evaluation in the root neighbourhood. Therefore, the absolute residual does not flush to zero and provides a sequence of iterates $x(k)$ that converges to $\overline{x}^*_-$ or $\overline{x}^*_+$ as long as the precision $\mathbf{u}$ is sufficient to improve the accuracy of $x(k)$, *i.e.*, while the RE criterion is not satisfied.

It follows that the splitting of the double root into two single roots does not come from the rounding errors of the corrected iteration (7). This is corroborated by the results of the interval Newton iteration in Section 4.5. We prove that the decimal to binary conversion of the coefficients of polynomial p is such that $\widehat{p}$ has two separate single roots in exact arithmetic. The following results come from symbolic computation with Maple. We represent the computed polynomial $\widehat{p}$ as $\widehat{p}(x) = \widehat{a}_3x^3 + \widehat{a}_2x^2 + \widehat{a}_1x + \widehat{a}_0$, where $\widehat{a}_i$ is the exact (rational) value of IEEE-754 single precision binary value of the a_i coefficient of p $(0 \leq i \leq 3)$. Using Sturm sequences, intervals I_- and I_+ include the considered exact roots of $\widehat{p}$ with, limiting to the maximum 8 significant digits of IEEE-754 single precision,

$$I_- = [0.4284\,9593; 0.4284\,9594] \quad \text{and} \quad I_+ = [0.4286\,4691; 0.4286\,4692].$$

We come back to previous floating point computations noting that both results computed in Section 4.4 satisfies $\overline{x}^*_- \in I_-$ and $\overline{x}^*_+ \in I_+$. Figure (2) represents computed $\widehat{p}$, corrected $\overline{p}$ and B_{E_C} bounds in x^* neighbourhood. It exhibits that $\overline{p}$ has two single separate roots in intervals $\widehat{I}_-$ and $\widehat{I}_+$ such that $\widehat{I}_- \subset I_-$ and $\widehat{I}_+ \subset I_+$. Furthermore, these two intervals I_- and I_+ are enclosed in the interval enclosures computed by the interval Newton method. Floating point experiments agree with the results of the symbolic computation.

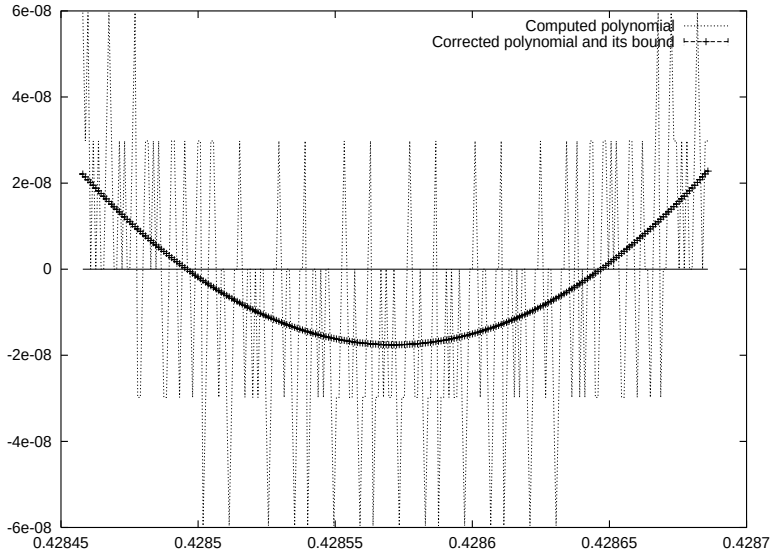

FIG. 2: The two separate roots of $\widehat{p}$ appear computing $\overline{p}$.

4.7 The Multiple Precision for interval Newton iteration

In this last series of experiments, we consider an arbitrarily increasing of the precision during interval Newton iteration as presented at the end of Section 3.3.

4.7.1 Polynomial coefficients are scalars

When the polynomial coefficients are the rounded to the nearest floating point numbers corresponding to the given decimal coefficients, the results for $\mathbf{u}_X = 10^{-6}$ and $\mathbf{u}_Y = 10^{-6}$ are three intervals containing one and only one single root.

In the following cases, interval endpoints differ by the last digit.

1. $[-1.6666\,6668\,0007\,92\underline{9}, -1.6666\,6668\,0007\,92\underline{1}]$: the isolated root $-5/3$ does not belong to this interval since the decimal to binary conversion has also modified this root ;
2. $J_- = [0.4284\,9593\,5588\,1302\,5201\,\underline{7}, 0.4284\,9593\,5588\,1302\,5201\,\underline{8}]$;
3. $J_+ = [0.4286\,4691\,1726\,057\underline{1}, 0.4286\,4691\,1726\,057\underline{3}]$.

These results are returned for computation with a maximal precision of 96 mantissa bits, *i.e.*, two doublings of the computing precision occurred to satisfy the stopping criteria. Intervals J_- and J_+ yield a relative accuracy on the roots of about 10^{-15}. We also observe an absolute accuracy on the residuals of 10^{-6} to 10^{-10}. The doubling precision strategy explains that these accuracies are better than the stopping criteria values $\mathbf{u}_x$ and $\mathbf{u}_Y$.

These experiments are another way to derive the symbolic results of Section 4.6 : indeed $J_- \subset I_-$ and $J_+ \subset I_+$. Other rounding modes for the decimal to binary conversion of the coefficients have been experimented. The corresponding polynomial has also two single roots when a downward rounding is applied and has no more roots in the neighbourhood of 3/7 with an upward rounding. This last case corresponds to an upward shift on Figure 4.6.

4.7.2 Polynomial coefficients are intervals

We terminate this experiment considering the interval polynomial whose coefficients are the intervals enclosing the coefficients. In this case, the result is the interval $[0.428\underline{4}, 0.428\underline{7}]$ for which no proof of existence or uniqueness can be given. Interval coefficients hence validate the two previous cases exhibited in the previous paragraph : a polynomial with two single roots and a polynomial with no root both belonging

to this interval polynomial. The lack of existence and uniqueness results observed with the single precision experiments (cf. Section 4.5.2) were thus not due to the lack of computing precision.

5 Conclusion

We have presented complementary applications of some automatic methods for rounding error analysis to a significant problem of finite precision computation. Such computation is not reliable when the problem to solve is ill-conditioned. In this case, floating point results are hard to interpret and may even yield wrong conclusions.

Automatic approaches are efficient tools that complement theoretical analysis. Finite precision effects are often subtle and merged with numerous and difficult other aspects. Let us cite the effects of condition, the underlined presence of singularities, data errors, truncation errors, algorithm stability, properties of floating point arithmetic, elementary functions faithfulness, library evolutions, compilers options, woolly specifications of programming languages, ... Therefore, automatic approaches cannot answer universally. They do not replace the knowledge of a numerical software expert but help him to confirm his intuition.

Acknowledgments : The author thanks Jean-Marie Chesneaux for the CADNA library and his help during its use.

Références

[1] J. H. Wilkinson, Rounding Errors in Algebraic Processes, Notes on Applied Science No. 32, Her Majesty's Stationery Office, London, 1963, also published by Prentice-Hall, Englewood Cliffs, NJ, USA. Reprinted by Dover, New York, 1994.

[2] N. J. Higham, Accuracy and Stability of Numerical Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[3] J. Vignes, A stochastic arithmetic for reliable scientific computation, Math. and Comp. in Sim. 35 (1993) 233–261.

[4] P. Langlois, Automatic linear correction of rounding errors, BIT 41 (3) (2001) 515–539.

[5] J.-M. Chesneaux, Stochastic Arithmetic and CADNA software, Habilitation à Diriger des Recherches Thesis, Université Pierre et Marie Curie, Paris, France, 1995, (In French).

[6] J.-M. Chesneaux, S. Guilain, J. Vignes, La bibliothèque CADNA : présentation et utilisation, Manual, Université P. et M. Curie, Paris, available at URL = `http://www-anp.lip6.fr/cadna/`, (in French) (November 1996).

[7] R. Moore, Methods and application of interval analysis, SIAM Studies in Applied Mathematics, 1979.

[8] A. Neumaier, Interval methods for systems of equations, Cambridge University Press, 1990.

[9] R. Baker Kearfott, Rigorous global search : continuous problems, Kluwer, 1996.

[10] A. Griewank, Evaluating derivatives. Principles and techniques of algorithmic differentiation, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[11] T. J. Dekker, A floating-point technique for extending the available precision, Numer. Math. 18 (1971) 224–242.

[12] D. E. Knuth, The Art of Computer Programming, Volume 2, Seminumerical Algorithms, 3rd Edition, Addison-Wesley, Reading, MA, USA, 1998.

[13] P. Langlois, F. Nativel, When automatic linear correction of rounding errors is exact, C.R. Acad. Sci. Paris, Série 1 328 (1999) 543–548, erratum in 328 :829, 1999.

[14] T. Braconnier, P. Langlois, From rounding error estimation to automatic correction with automatic differentiation, in : G. Corliss, C. Faure, A. Griewank, L. Hascoet, U. Naumann (Eds.), Automatic Differentiation : from Simulation to Optimization, Springer-Verlag, 2001, pp. 331–341.

[15] M. Daumas, J.-M. Muller (Eds.), Qualité des calculs sur ordinateur, Masson, 1997.

[16] G. Hanrot, V. Lefèvre, F. Rouillier, P. Zimmermann, The MPFR library, URL = `http://www.mpfr.org` (2001).

[17] N. Revol, F. Rouillier, The MPFI library, URL= `http://www.ens-lyon.fr/~nrevol` (2001).

[18] J. M. McNamee, A bibliography on roots of polynomials, J. Comput. Appl. Math. 47 (3) (1993) 391–394, (Supplementary WWW bibliography at URL = `http://www.elsevier.nl/locate/cam`.).

[19] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, 2nd Edition, Springer-Verlag, New York, 1993.

[20] G. Dahlquist, Å. Björck, Numerical Methods, Prentice-Hall, Englewood Cliffs, NJ, USA, 1974, translated by Ned Anderson.

[21] L. B. Rall, Convergence of the Newton process to multiple solutions, Numer. Math. 9 (1966) 25–37.

[22] G. Peters, J. H. Wilkinson, Practical problems arising in the solution of polynomial equations, J. Inst. Maths Applics 8 (1971) 16–35.

[23] J. Vignes, Zéro mathématique et zéro informatique, La vie des Sciences, C.R. Acad. Sci. Paris 4 (1) (1987) 1–13, (In French).

[24] J.-M. Chesneaux, J. Vignes, Les fondements de l'arithmétique stochastique, C. R. Acad. Sci. Paris Sér. I Math. 315 (13) (1992) 1435–1440.

[25] P. Van Hentenryck, L. Michel, Y. Deville, Numerica, a modeling language for global optimization, MIT Press, 1997.

[26] N. Revol, Newton iteration using multiple interval arithmetic : the univariate case, in : Linear Algebra and Arithmetic, Rabat, Morocco, 2001.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399